PLAGO: A SYSTEM FOR PLAGIARISM DETECTION AND INTERVENTION IN MASSIVE COURSES

A Thesis
Presented to
The Academic Faculty

by

Carmine T. Guida

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Computer Science

Georgia Institute of Technology
August 2019

PLAGO: A SYSTEM FOR PLAGIARISM DETECTION AND INTERVENTION IN MASSIVE COURSES

Approved by:

Dr. David Joyner, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Ada Gavrilovska
School of Computer Science
Georgia Institute of Technology

Dr. Alessandro Orso
School of Computer Science
Georgia Institute of Technology

Date Approved: July 18, 2019

To Mia, Meredith, Lily and the Cast and Crew of LucyLabs

ACKNOWLEDGEMENTS

The Online Master of Science in Computer Science (OMSCS) program made it possible for me to pursue a degree from almost 1,000 miles away and maintain my constantly shifting work and life schedule. Dr. David Joyner's courses (as a student and as his Teacher Assistant) were eye opening and set me on a trajectory to start a new career as a Professor here in New York City. I am grateful for the OMSCS program and the opportunities, challenges and conversations with Dr. Joyner that lead to me developing this system and pursuing the Thesis option.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
CRUD	Create, Read, Update and Delete
FERPA	Family Educational Rights and Privacy Act
JSON	JavaScript Object Notation
LMS	Learning Management System
MOSS	Measure of Software Similarity
NLTK	Natural Language Toolkit
OMSCS	Online Master of Science in Computer Science
PDF	Portable Document Format
TA	Teacher Assistant
URL	Uniform Resource Locator

SUMMARY

This thesis covers research into various extrinsic and intrinsic plagiarism detection methods such as n-gram comparisons as well as stylometry using natural language processing techniques. This thesis also provides a technical description and review of the development of Plago which took place over three semesters. Finally, results of plagiarism detected in student assignments are presented as well as a discussion of potential further work and research.

Plago is as a tool for Instructors and TAs, especially in massive (200+ students) courses. Plago imports students' essay style PDF assignment submissions directly from Canvas LMS and prepares them for scanning for plagiarism. Comparisons against a corpus of tens of thousands of existing papers takes a few minutes and flags potential plagiarism based on match percentages. Additionally, Plago provides evidence of plagiarism in way of a side by side comparison of papers with highlighting on matching text between the two papers.

CHAPTER 1. INTRODUCTION

Plagiarism is an ongoing issue for educational institutions and has been referred to as a problem that "won't go away" [1]. Plagiarism can be defined as when students copy words or ideas from a published source straight into their assignment without acknowledging the source [2]. Hannabuss referred to plagiarism as "the unauthorized use or close imitation of the ideas and language/expression of someone else" [3]. Another reason plagiarism is a problem is that it is meant to deceive the reader or potentially the grader of an assignment [4, 5].

1.1 Different Types of Plagiarism

There are different types and severities of plagiarism. Collusion or over-collaborating between students can also be seen as a form of plagiarism. In the context of written assignments, Carroll and Zettering described the kinds of student plagiarism as: copying words and ideas from published sources, copying from other students and working too closely with other students [6]. Martin provides an extended list of types of plagiarism [7]:

- Word-for-word plagiarism: also known as copy-paste plagiarism.
- Paraphrasing plagiarism: changing some, but not enough words.
- Plagiarism of secondary sources: referring to an original source but never having read the source.
- Plagiarism of the form of a source: using the structure of an argument without acknowledging the source.

- Plagiarism of ideas: someone else's idea is used without referring to the source.
- Plagiarism of authorship: putting one's name on someone else's work.

Self-plagiarism can be seen as another form of deception. Collberg et al. describe self-plagiarism as "the use of one's own previously published materials in the creation of a new published material without crediting the previous as a source" [8]. Hexham preferred to call self-plagiarism "recycling fraud" when "no indication that the work is being recycled and an effort is made to disguise the original text" [9].

1.2 Penalties

Mawdsley referred to plagiarism as, "a form of academic dishonesty that speaks to the very heart of higher education" [10]. Educational institutions typically have policies about academic integrity with a statement on plagiarism and guidelines for violations. Penalties for academic misconduct can vary between institutions. The Office of Student Integrity at Georgia Tech offers sanction guidelines which range from a zero on an assignment, to failure in a course and potential expulsion [11].

1.3 Goals of Plagiarism Detection Systems

Determining if students' papers contain plagiarism is a time-consuming task and may be near impossible with a large class of 200 or more students. With regards to a manual approach to plagiarism detection, Niezgoda and Way referred to it as "labor intensive, requiring detailed, on-screen reading and re-reading of each paper" [12]. Automated plagiarism detection systems are a tool to reduce the work load and burden on

Instructors and other staff. These systems are not grading assignments but, rather, flag assignments or "can play a role in fostering proper acknowledgement practice by alerting teachers and students to passages that are incorrectly quoted or insufficiently acknowledged" [13]. Additionally, plagiarism detection systems may aid in the presentation of evidence for academic integrity cases. Finkel et al. developed software which would "build a plagiarism or a cheating case leading to a disciplinary process" [14].

1.4 Summary of Detection Methods

There are two major categories for plagiarism detection: extrinsic and intrinsic. Extrinsic plagiarism detection requires a corpus of original reference sources [15, 16]. A document is turned into fragments or n-grams and compared against the existing documents in the corpus.

Intrinsic plagiarism detection refers to comparing a document against itself. Human readers may identify suspicious passages within a document without having a library of reference documents in mind [16]. Changes in writing style (stylometric features) within a document or between a document and the author's previous work may indicate plagiarism.

CHAPTER 2. EXISTING RESEARCH AND SYSTEMS

2.1 Extrinsic Plagiarism Detection

Systems based on extrinsic plagiarism detection use a corpus of reference documents for comparisons. Documents are divided into blocks of words (N-grams) and may involve extracting stop words and stemming.

2.1.1 N-grams

Barrón-Cedeño and Rosso explained, "due to the fact that a plagiarized sentence could be made of fragments from multiple parts of an original document, the reference documents should not be split into sentences, but simply into n-grams" [15].

An n-gram is a sequence of words of length n. Given the sentence "Computer Science is lots of fun", example bigrams (2-gram) would be *Computer Science* and *Science is*. N-grams are built as a "sliding window" across the text. For example, when using trigrams (3-gram) the resulting list would be: *Computer Science is*, *Science is lots*, *is lots of* and *lots of fun*.

Different sizes of n-grams are used depending on the desired results. Trigrams give better precision however bigrams offer better recall [15]. Typically, trigrams are used [17]. After finding candidate documents using bigrams or trigrams some systems perform further comparisons using larger n-grams. Long blocks of identical words are strong indicators of plagiarism [18].

2.1.2 Stop Words

Stop words are common words such as *is*, *a* and *the*. These words indicate the structure of a sentence and the relationships between the concepts presented, but do not have any meaning on their own [18]. Stop words are often removed in plagiarism detection systems [19, 20]. Appendix A contains the list of stop words used by Plago. Removing stop words would reduce our previous example "Computer Science is lots of fun", to *Computer Science lots fun*. Following stop word removal, the remaining text would be turned into a trigram list of *Computer Science lots* and *Science lots fun*.

2.1.3 Structural Matching

Students may attempt to deceive the reader (or a plagiarism detection system) by changing only the "important" words. For instance, a sentence such as "The Professor is in the lecture hall" might be changed to, "The Instructor is in the classroom." Removing the stop words for each example would yield *Professor lecture hall* and *Instructor classroom*. This would clearly elude a simple plagiarism detection system. Stamatatos proposed a method based only on structural information where instead of removing stop words, they are kept, and the remaining words are eliminated [20]. With the previous example, retaining the stop words and removing the other words would give "the is in the" for both blocks of text and indicate a structural match. Hoad and Zobel also found stop words "may be a useful indicator of coderivation" and "plausible that they occur in similar frequencies in documents that are copied" [18].

2.1.4 Stemming

Another text processing step in extrinsic plagiarism detection systems is to use a stemming algorithm [21] on regular words [19, 22]. The words “computer, computation, computational” would be reduced to the root *comput* and words such as “education, educational, educator” would be reduced to *educat*. Using root word in pattern matching provides a much better effectiveness in information retrieval [23].

2.2 Intrinsic Plagiarism Detection

Extrinsic plagiarism detection is only effective when source documents are available. It would be unfeasible to store the entirety of the World Wide Web as a reference corpus or to perform an Internet search on every sentence for every student. Intrinsic plagiarism detection requires comparing a document only against itself. Suspicion of plagiarism is based exclusively on irregularities or inconsistencies within a document that are stylistic in nature [24].

2.2.1 Stylistic Measures

A writing style profile can be quantified by various properties. Zu Essen and Stein define five categories of stylometric features [16]:

- Text statistics, which operate at the character level
- Syntactic features, which measure writing style at the sentence-level
- Part-of-speech features to quantify the use of word classes
- Closed-class word sets to count special words
- Structural features, which reflect text organization

Blocks of text within a document can be examined for deviations from the overall writing style. Such passages can be used as a starting point for Internet searches for the original source or to be set aside for inspection by a human [16].

2.3 Existing Systems

There are several existing commercial and non-commercial plagiarism detection systems available. Some offer a software as a service model with a global corpus of documents, while others are run locally with private corpora.

2.3.1 MOSS

MOSS (for Measure of Software Similarity) is an automatic system for determining the similarity of programs [25]. MOSS is used for detecting code plagiarism in various computer languages and is a standard tool for programming courses [26]. Similar to systems for detecting plagiarism in written assignments, MOSS provides similarity scores to be checked manually by course staff [27]. Control structures, identifiers, literals and other parts of submitted source code are converted into tokens. The main advantage of such an approach is that it negates all lexical changes and a good token set can also reduce the efficacy of many structural changes [28].

Unfortunately, MOSS is not an open source system. MOSS is provided as a service and student assignments must be uploaded to a central server. There are various scripts for uploading as well as viewing the results for several platforms, however individuals are unable to customize the algorithm or add additional programming languages.

2.3.2 Turnitin

Turnitin is a commercial subscription-based plagiarism detection system for written assignments offering integration with learning management systems such as Canvas and Blackboard. In the "Technical Review of Plagiarism Detection Software Report", Bull Et al. describes Turnitin as detecting "material copied from the Internet as well as collusion between students through cross-checking of submitted essays against one another and against an in-house database of texts" [29]. An originality report is provided for each document with links to matching material found on the Internet [30].

Turnitin can be an attractive solution however several issues make it problematic for academic settings:

- Students can potentially game the system by uploading and checking their papers with Turnitin outside of their course.
- Papers become part of a global data set. Effectively, Universities are paying to give data and improve Turnitin. Additionally, there may be FERPA (privacy) issues with students' information in their papers.
- Turnitin is a third party and therefore requires opt-in by students with each paper submission. Given this requirement, Universities are unable to submit previous documents from the student or former students.

CHAPTER 3. TECHNICAL DETAILS AND METHODS

Plago was developed over three semesters. The implementation was designed to use resources from the College of Computing at Georgia Institute of Technology (Georgia Tech). These resources were freely available to the researcher as a student. Plago is up and running and can be accessed by Instructors and TAs after an account has been added for them. Plago uses the Georgia Tech single sign-on system and (at the time of writing) can be accessed via the following URL: http://plago.cc.gatech.edu

3.1 Architecture

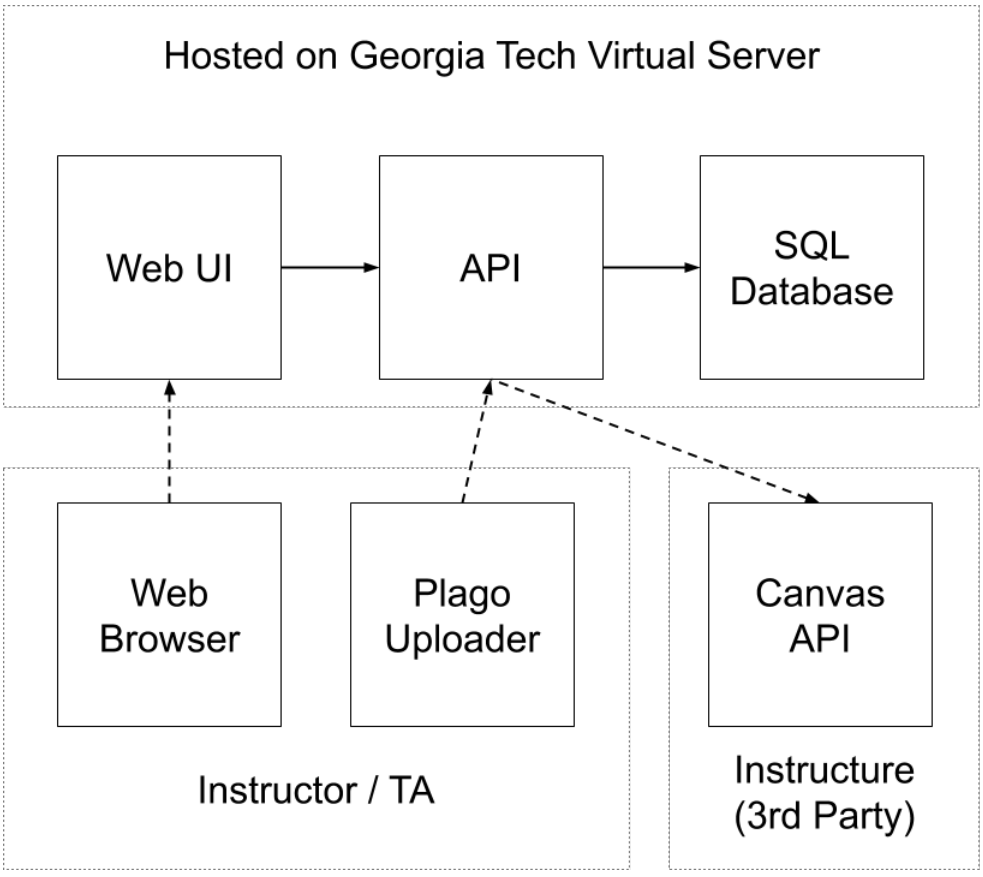

Figure 3.1: Architecture

Figure 3.1 shows the overall architecture for Plago. The system is hosted in a virtual server environment at Georgia Tech. The operating system is Windows Server 2012 R2 Standard, the database engine is Microsoft SQL Server 2017 (Web Edition) and the user interface (web content) as well as the API are delivered via Internet Information Server (IIS) version 8.5. The UI is based on HTML, CSS and JavaScript which makes requests to the API. The API is written in Python and uses the Flask framework. The API pulls papers from Canvas LMS utilizing the Canvas API. Finally, all data (papers, statistics, etc.) are stored inside the database with most of the n-gram comparisons being performed by stored procedures.

3.2 UI and Feature Overview

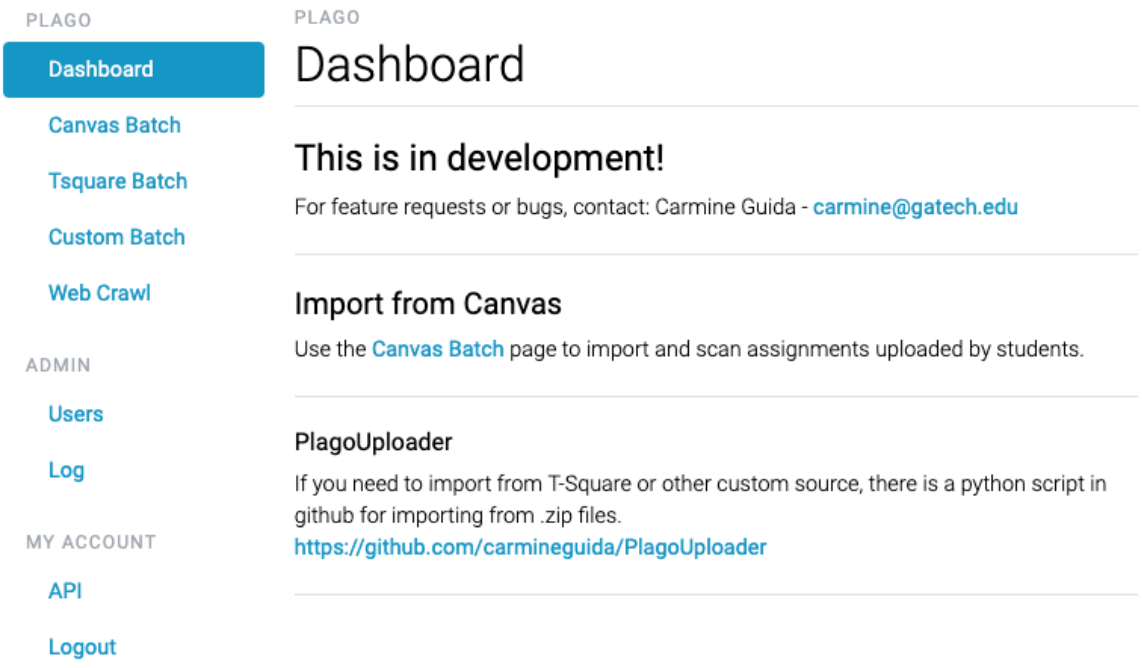

Figure 3.2: Web Dashboard

Instructors and TAs use Plago mostly through the web-based user interface. After logging into the system, users are presented with the dashboard (Figure 3.2). The navigation provides access to the following:

- Canvas Batch: For queuing batches to be imported from Canvas LMS as well as monitoring the progress of importing and processing of batches.
- T-Square Batch: Before migrating to Canvas in 2018, Georgia Tech used an LMS known as T-Square. Using the Plago Uploader tool (described in 3.5.1) users can upload exported archives from T-Square.
- Custom Batch: For monitoring batches from Plago Uploader that are not assignments from Canvas or T-Square.
- Web Crawl: Used for queuing and monitoring of importing web pages from the CommonCrawl.org public dataset (described in 3.5.2).
- Admin Options
    - Users: For giving users access to the Plago system.
    - Log: Used to review user logins, start and end of batches and other system information.
- API: This link provides course staff with an API key for using Plago Uploader or their own custom tools to access the Plago API on their behalf.

3.3 Plago API

Table 3.1: Plago API Examples

URL	JSON
/api/canvas_integration_add	{ "course_id":"8700", "assignment_id":"6230" }
/api/scan_add	{ "batch_id:":2112, "course_match_type":1, "course_match_text":"", "assignment_match_type":1, "assignment_match_text":"" }

The Plago API is used for performing CRUD operations on the database as well as executing various processes. The Plago API is accessed via JavaScript in the Web UI

and additionally by external tools such as Plago Uploader or a custom tool developed by an Instructor or TA. Table 3.1 shows examples of requests that can be made to the Plago API. The URL determines the method to be performed. All data is sent and received in JSON format. The first example queues an assignment from a course in Canvas for importing. The second queues a batch that was imported to be scanned for plagiarism.

3.4 Importing Papers

All of the student submissions for an assignment are imported and processed to be available for scanning for plagiarism. Assignments may be imported using the Canvas API or extracted from other sources and uploaded with the Plago Uploader tool. Plago can process .PDF, .TXT and .MD files.

3.4.1 Canvas Integration

Canvas LMS has an API for accessing courses, assignments, students, and more. Users can generate an API key inside of Canvas and then store that key inside of Plago. With the Canvas API key, Plago can download data from Canvas on behalf of that user. An Instructor or TA can navigate to the Canvas Batch area in Plago and queue an assignment to be imported. Figure 3.3 shows an example of selecting a course and Figure 3.4 shows assignments that can be selected for importing. Multiple assignments can be queued at one time. A background process runs every 5 minutes which looks for pending batches to be imported from Canvas and into Plago for processing.

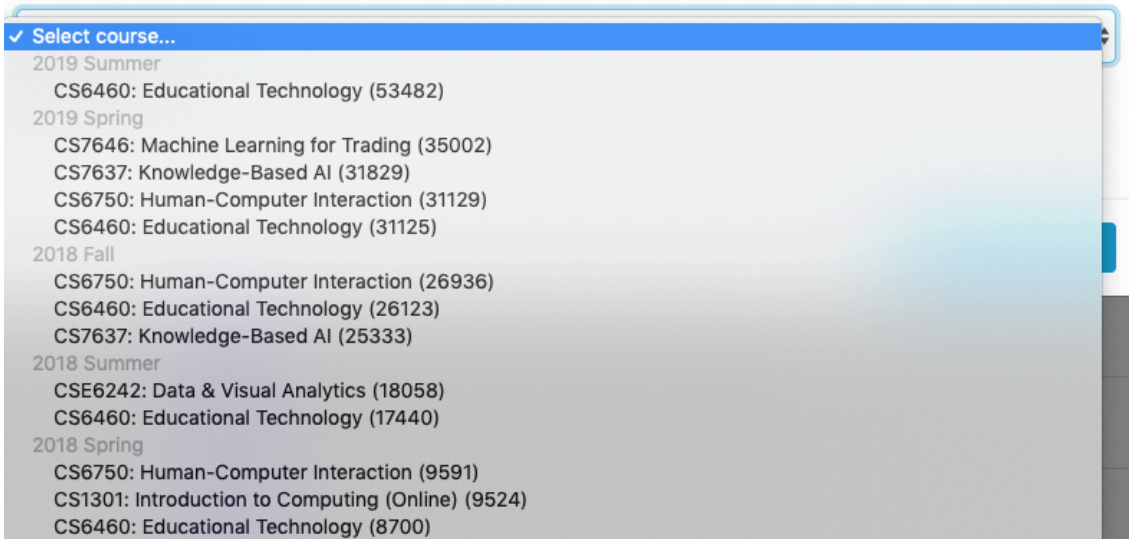

Figure 3.3: Canvas Integration Course Selection

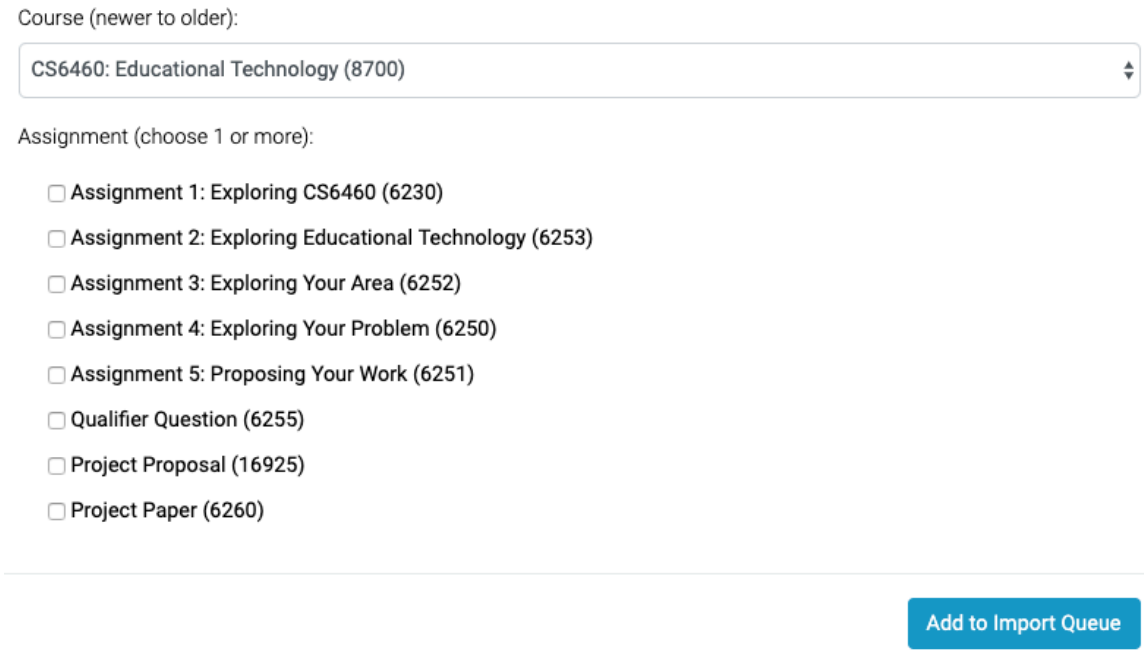

Figure 3.4: Canvas Integration Assignment Selection

3.4.2 Processing

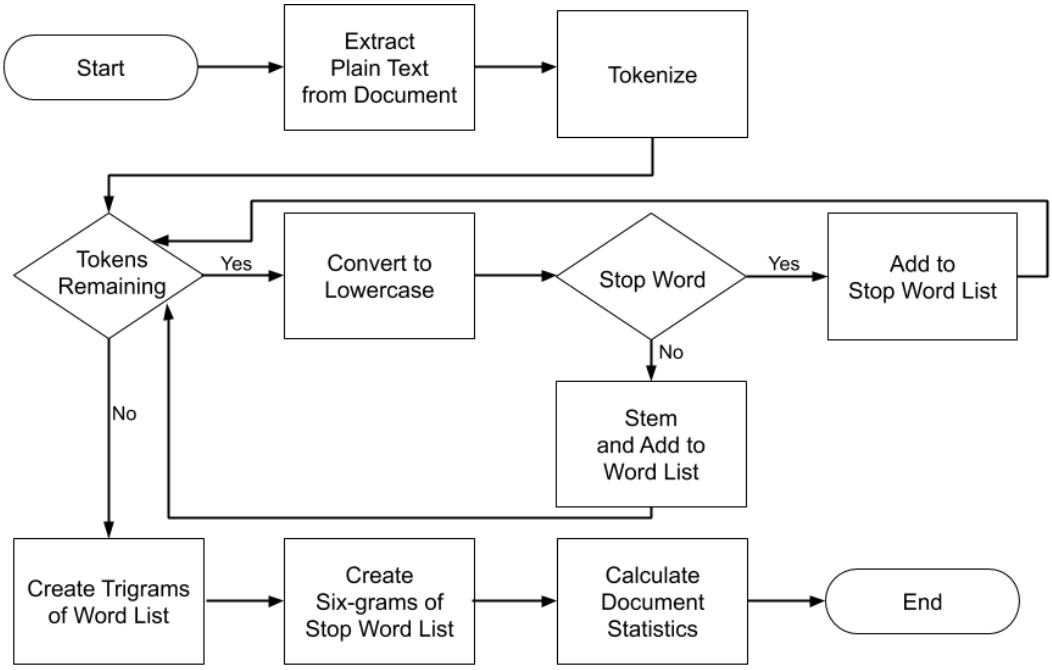

Figure 3.5: Text Processing

Figure 3.5 details how Plago processes text for later scanning and analysis. First, the plain text of the document needs to be extracted. For PDF files, Plago uses the PDFMiner Python library. The text is then tokenized (split into individual words) by the NLTK [31]. Each word is converted to lowercase. If a token is a stop word, it is simply added to a list. Regular words are "stemmed" using the NLTK [31] Porter [21] stemmer function and added to a separate list. Stemming reduces words to their root, potentially increasing matches even when small changes are made and has an added benefit of reducing database storage needs.

After the regular word and stop word lists are created. A list of trigrams [15, 17] is built from the regular word list and a list of 6-grams is built from the stop word list (used for structural matching). The original text, trigram list, 6-gram list and PDF metadata are

stored in the database. Stylistic statistics [16] about the text is calculated and also stored in the database. These statistics are intended to be used for intrinsic analysis. At the time of writing, intrinsic analysis is not in use for this project, however the list of collected statistics can be found in Appendix B.

3.5 Scanning for Plagiarism

After the text of a document is processed, n-grams are built and stored as part of the corpus. The n-grams from the new document are ready to be compared against n-grams from other student submissions for potential plagiarism.

3.5.1 Scan Batch Settings

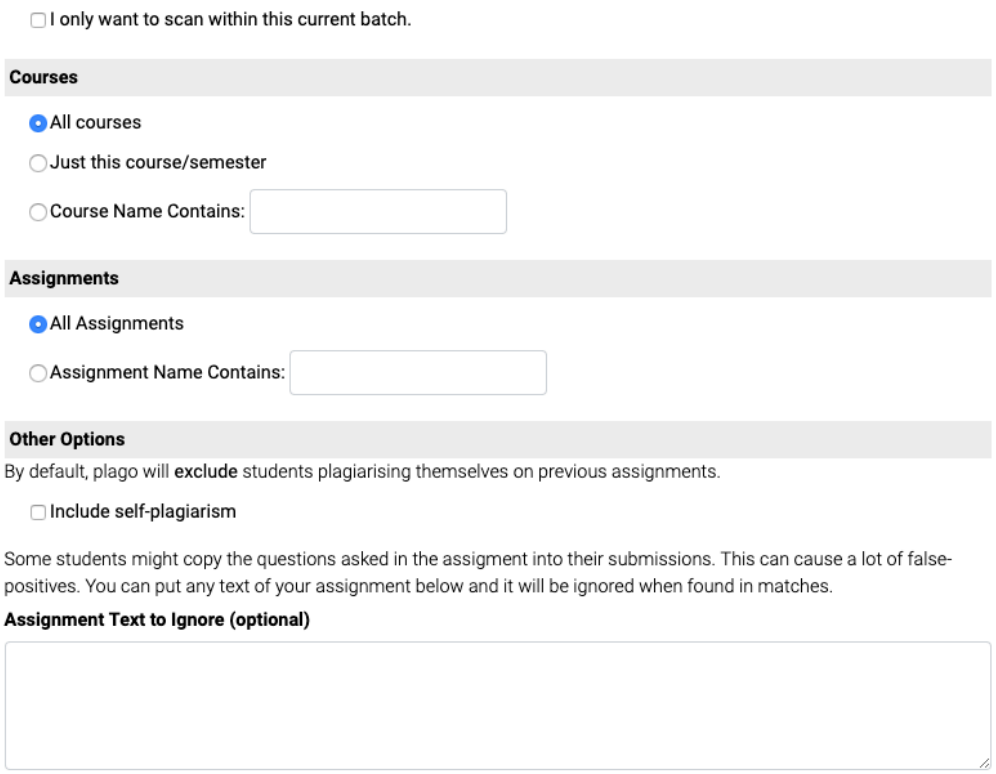

Figure 3.6: Scan Batch Settings

An Instructor or TA can queue the scanning of a batch of imported and processed student papers. A scan can be configured to the user's needs, and multiple styles of scans can be created per assignment batch. Figure 3.6 shows all of the options available. By default, Plago will scan against all papers in the corpus. This can be changed to only include the current batch or filter by a particular course or assignment. For example, a batch can be compared against all semesters of CS6460 and/or all assignments named "Midterm Paper 1." Users can choose to include students plagiarizing from their own previous assignments. Additionally, users can copy and paste text to be ignored (for example, the questions in an assignment that students might include in their papers).

3.5.2 Scanning Process

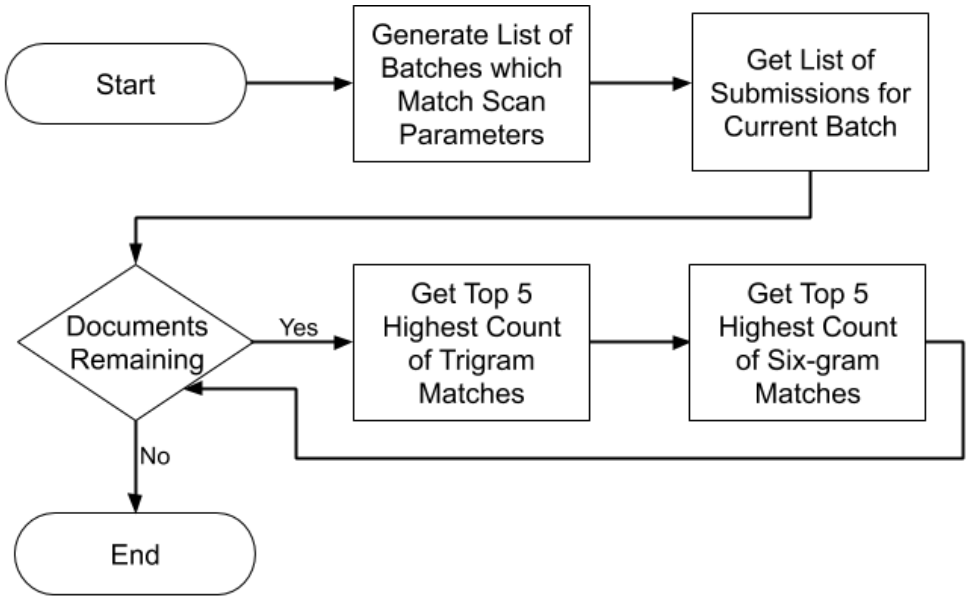

Figure 3.7: Scanning Process

Figure 3.7 details the process of scanning an imported batch for plagiarism. First a list of batches that match the scan parameters (mentioned in 3.5.1) is generated. Plago then iterates through the batch being scanned. For each document, Plago gets a list of the

top 5 candidates for plagiarism based on the count of matching trigrams from other documents in the list of batches to scan against. Plago also builds another list of the top 5 candidates for each document based on 6-grams of stop words for structural matches. The scan results are listed in highest match order descending.

3.5.3 Scan Results

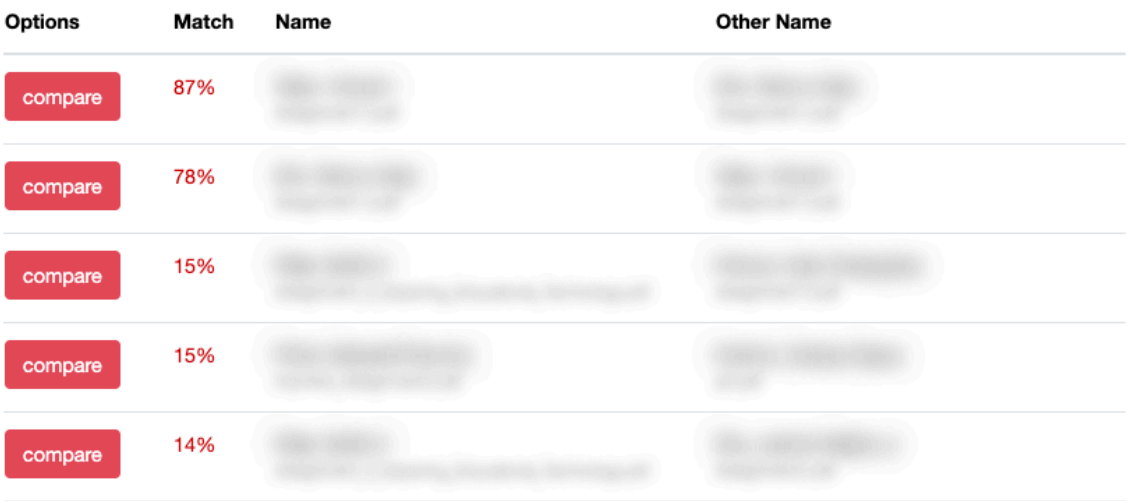

Figure 3.8: Scan Results (Student Names Redacted)

Figure 3.8 shows the results of a plagiarism scan (with the students' names redacted) for an assignment that was imported into Plago. The match percentage refers to the count of trigrams from the source document that were found in the other document. The document in the first row of Figure 3.8 (with an 87% match) was a 992-word paper with 609 trigrams. 532 of those trigrams were found in another student's paper.

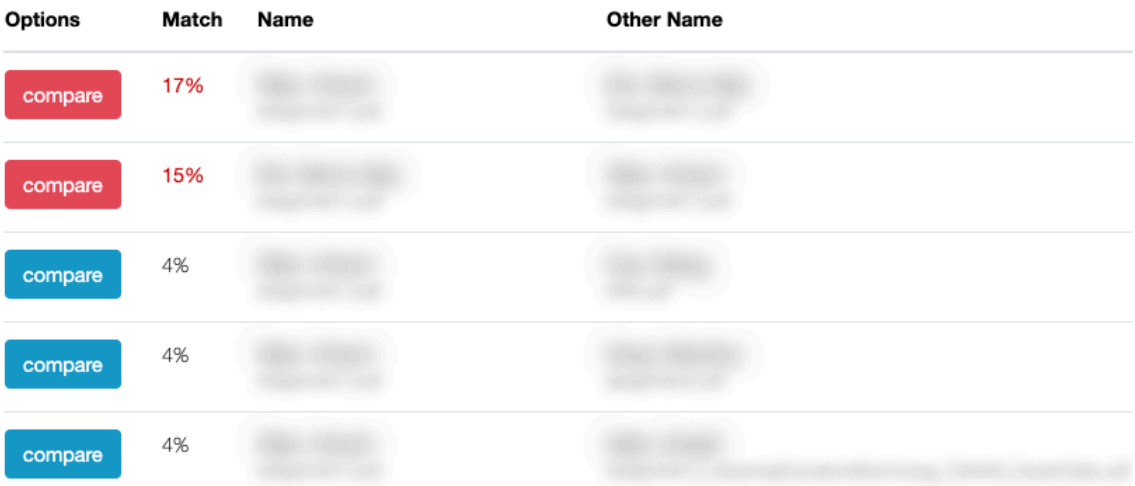

Figure 3.9: Structural Scan Results (Student Names Redacted)

Figure 3.9 is an example of a structural match comparison based on 6-grams of stop words. Papers with 10% or higher match may be a candidate for plagiarism with an attempt to change the "important words" while leaving the overall structure of the document intact [20].

3.5.4 Document Comparison

Figure 3.10: Side-by-Side Comparison (Text Redacted)

Figure 3.10 shows an example side-by-side comparison (with text redacted) of two student assignments. Colors are used to highlight matches between documents. This view helps the course staff review the documents and determine if plagiarism occurred.

3.6 Other Tools and Features

Early in the development of Plago, the following tools were developed for importing text from other sources. As Plago evolved, the focus was moved to Canvas integration and comparisons against previous student assignments.

3.6.1 Plago Uploader

Plago Uploader is a publicly available Python script which uploads files into Plago using the Plago API. This was the initial method for uploading files into Plago during its development before the integrated Canvas API discussed in 3.3.1 was created.

Plago Uploader can connect to Canvas, open archives from T-Square and be customized as needed by an Instructor or TA. At the time of writing, the source code for Plago Uploader can be accessed at the following URL:
https://github.com/carmineguida/PlagoUploader

3.6.2 Web Importer

The primary corpus of text for Plago is based on current and past students' assignments. Students may plagiarize from other sources such as articles from Web pages. Common Crawl is a non-profit organization which offers a public dataset containing petabytes of data from web pages. Plago can download html from the Common Crawl dataset and extract the text using the BeautifulSoup Python library.

Plago users can help build the corpus of web pages for comparing against student assignments by specifying an Internet domain name, or by a URL prefix. For example, "*.cc.gatech.edu" would import (at the time of writing) text from over 4,000 web pages. A query such as "https://en.wikipedia.org/wiki/computational*" would add all Wikipedia articles beginning with the word computational such as "Computational Thinking", "Computational Problem" and "Computational Journalism."

CHAPTER 4. **RESULTS**

The following are the results over the development of Plago and subsequent beta test period. Development began during the Fall 2018 semester with initial scans and detected plagiarism occurring towards the end of the semester. During the Spring 2019 semester, text pre-processing and comparison algorithm were enhanced. The web user interface was developed as well as Canvas integration for importing batches of assignments. At the time of writing (Summer 2019 semester), course staff are able to access Plago to import documents and generate reports on match percentages between student submissions without the researcher's involvement.

4.1 Usability

Plago's corpus was built from an initial import of historical student submissions from 3 courses over 4 semesters. The corpus contains 97 assignments and over 23,000 documents. Queuing a new batch (Figure 3.3 – 3.4) happens quickly, however the background process (Figure 3.5) which imports the documents to the corpus can take hours depending on the number and complexity of submitted PDF files. Scheduling a scan (Figure 3.6) is also quick for the user. The comparison report (Figure 3.8 – 3.9) is generated within 5 to 20 minutes. The side by side document comparison (Figure 3.10) is generated within 2 minutes and is cached for instant future access.

4.2 Plagiarism Detected

Within a window of 6 months, approximately 20 students were caught plagiarizing which resulted in cases submitted to Georgia Tech's Office of Student

Integrity. These instances of plagiarism caught by Plago were undetected by the course staff. Given the need to maintain privacy, details about the students, courses and assignments are not available in this thesis.

4.3 Repeat Offenders

One of the motivations for creating Plago was to detect plagiarism early and deter further plagiarism from students. Unfortunately, given the relatively small window of time and the researcher's access to only a few courses, it is unknown if the students caught by Plago plagiarized on later assignments either within the course they were enrolled in or other courses in the program.

4.4 Benefits vs. Existing Systems

Existing plagiarism detection systems (such as Turnitin) are offered as a "software as a service" model. This forces students to agree to give their data to a third party and to have their papers added into a global database. Plago is a "homegrown" solution residing on a virtual server within Georgia Tech's campus. This maintains the privacy of students, keeps control of the students' data with the University and also allows for importing of historical assignments as needed.

CHAPTER 5. CONCLUSIONS

5.1 Discussion and Future Work

Limitations on time, hardware resources, the types of assignments analyzed, and the detected plagiarism during the development of this project have presented several potential areas of future research and work.

5.1.1 Avoiding Bias in Algorithms

The researcher explored intrinsic plagiarism detection using various statistics about writing style including stop word ratio, rarity of vocabulary words and syllables per word. A profile containing the mean, median and standard deviation was stored for each paper. During analysis, Plago compares the statistics on a sentence by sentence level against the overall profile for the paper. Outlier sentences at 2 standard deviations and above were flagged as potential sources of plagiarism. This led to several false positives and was especially biased against students with English as a second language. Properly cited quotes from sources were interpreted as severe spikes in vocabulary rarity and sharp reductions in stop word usage. Given the bias of this algorithm and time constraints for the project, this method was removed.

5.1.2 Fingerprint Methods

A fingerprint or profile of the student's writing style can be made using the previously mentioned statistics as well as parts-of-speech and punctuation usage. The fingerprint could be compared against a student's assignment submissions within a course, across different courses and even previous semesters to look for drastic change in style. A substantial change could indicate plagiarism or the usage of a paper writer for hire.

5.1.3 Web Page Corpus

The discussed intrinsic plagiarism detection methods flag a paper as suspicious, however they do not provide *evidence* of plagiarism. The TA or Instructor would need to perform online searches to find the original source that was copied. Plago has the capability to store web pages downloaded from the Common Crawl public dataset. The intention behind this feature was to collect text from sources containing subject matter common for student assignments. For instance, Plago could store text from Wikipedia articles and homework help websites about Machine Learning, Artificial Intelligence, Ravens Progressive Matrices and other common Computer Science topics. These web pages would act as an additional corpus of text (and evidence) for plagiarism detection. This feature is not currently being utilized as it requires people knowledgeable in several fields to enter potentially hundreds of URLs to build an effective secondary corpus.

5.1.4 Optimal n-gram sizes

For n-gram size, Plago uses N=3 for content similarity and N=6 for structural matching. Using trigrams for content was based on research into existing systems. Most of the systems discussed in this thesis used trigrams as a basis for their detection algorithms. Some systems have found success with casting a wider net using bigrams, others used four-grams (and higher) in order to decrease the number of false positives. Using a 6-gram for structural matching was derived by the average stop word count per sentence and approximating two sentences worth of stop words. The average stop words per sentence was based on student papers available to the researcher early in the development of the system (a single course in the OMSCS program). The choice of n-gram sizes may be heavily biased towards Computer Science papers and further research would need to be performed to find optimal settings for other topics and grade/writing levels.

5.1.5 Storage Optimizations

A class of 500 students may have 8 assignments throughout the semester. This class would add 4,000 papers to the corpus of texts. Throughout its lifetime in the system, a paper may be compared against other papers (within the same course and other courses) thousands of times. To facilitate fast scanning of documents, Plago stores the n-grams for content and structural comparison during the import process. Given an average paper length of 2,000 words, a single paper will generate approximately 1,900 additional rows of data in the database. With the current implementation, a single semester of a massive course stored in Plago adds 7.6 million rows of data. Data storage and backup will quickly become an issue. Further research needs to be performed in ways to trim down the amount of data stored per paper. For example, the system could ignore the beginning and end of papers or use a sampling method to store portions of the paper. A potential method to reduce storage requirements would be to purge or archive papers that were never used as a source for copying by other students, leaving only likely candidates for copying in the corpus.

5.2 Conclusions

During the development and beta testing of Plago, instances of plagiarism by students were found which were previously undetected by the course staff. The students in these courses where not informed that a plagiarism detection system was being used. Additionally, there were no announcements to the other students when their fellow students were caught plagiarizing. Further studies may need to be performed about the effects of informing students about the usage of plagiarism detection systems and when their fellow students (while maintaining privacy) have been caught. This information as well as further education for returning students about plagiarism could potentially reduce first time plagiarism. However, the knowledge of the usage of plagiarism detection systems

(especially systems which are open source) could create an arms race between the detection systems and students becoming more sophisticated in how they mask copying other works. Overall, in order to *eliminate* plagiarism, a more holistic approach may be needed that not only detects plagiarism, but also prevents it.

APPENDIX A. STOP WORDS

Stop words are common words in a language which are typically removed before processing. All of the incoming text in Plago is converted to lowercase first.

A.1 NLTK Stop Words

The following is a list of English stop words as provided by the NLTK [31]:

i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, can, will, just, don, don't, should, should've, now, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't

A.2 Additional Stop Words

The following stop words were added by the researcher.

cs, s, t, d, ll, m, o, re, ve, y, 's, 't, 'm

APPENDIX B. DOCUMENT STATISTICS

During the processing step, Plago calculates and stores statistics for each document. These statistics are intended to be used for intrinsic plagiarism detection.

B.1 List of Collected Statistics

Word Count

Character Count

Stop Word Ratio

Sentence Word Count (Mean, Median and Standard Deviation)

Sentence Stop Word Count (Mean, Median and Standard Deviation)

Word Length (Mean, Median and Standard Deviation)

Syllables per Word (Mean, Median and Standard Deviation)

Vocabulary Word Rarity (Mean, Median and Standard Deviation)

REFERENCES

[1] Paldy, L. G. (1996). The Problem That Won't Go Away: Addressing the Causes of Cheating. *Journal of college science teaching*, *26*(1), 4-6.

[2] Carroll, J., & Zetterling, C. (2009). Guiding students away from plagiarism. *KTH Learning Lab*.

[3] Hannabuss, S. (2001). Contested texts: issues of plagiarism. *Library management*, *22*(6/7), 311-318.

[4] Statement on Plagiarism. (1989). *Academe, 75*(5), 47-48.

[5] Clough, P. (2003). Old and new challenges in automatic plagiarism detection. National UK Plagiarism Advisory Service.

[6] Carroll, J., & Zetterling, C. (2009). Guiding students away from plagiarism. *KTH Learning Lab*.

[7] Martin, B. (1994). Plagiarism: a misplaced emphasis. *Journal of Information Ethics*, *3*(2), 36-47.

[8] Collberg, C. S., Kobourov, S. G., Louie, J., & Slattery, T. (2003, November). SPLAT: A System for Self-Plagiarism Detection. In *ICWI* (pp. 508-514).

[9] Hexham, I. (2005). The plague of plagiarism: Academic plagiarism defined. *Calgary: University of Calgary*.

[10] Mawdsley, R. D. (1986). Plagiarism problems in higher education. *JC & UL*, *13*, 65.

[11] Georgia Institute of Technology, The Office of Student Integrity (n.d.). Retrieved from http://osi.gatech.edu/content/possible-outcome-process

[12] Niezgoda, S., & Way, T. P. (2006, March). SNITCH: a software tool for detecting cut and paste plagiarism. In *ACM SIGCSE Bulletin* (Vol. 38, No. 1, pp. 51-55). ACM.

[13] Martin, B. (2004). Plagiarism: policy against cheating or policy for learning?.

[14] Finkel, R. A., Zaslavsky, A., Monostori, K., & Schmidt, H. (2002, January). Signature extraction for overlap detection in documents. In *Australian Computer Science Communications*(Vol. 24, No. 1, pp. 59-64). Australian Computer Society, Inc..

[15] Barrón-Cedeño, A., & Rosso, P. (2009, April). On automatic plagiarism detection based on n-grams comparison. In *European Conference on Information Retrieval* (pp. 696-700). Springer, Berlin, Heidelberg.

[16] Zu Eissen, S. M., & Stein, B. (2006, April). Intrinsic plagiarism detection. In *European Conference on Information Retrieval*(pp. 565-569). Springer, Berlin, Heidelberg.

[17] Lyon, C., Barrett, R., & Malcolm, J. (2004). A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector. *Plagiarism: Prevention, Practice and Policies*.

[18] Hoad, T. C., & Zobel, J. (2003). Methods for identifying versioned and plagiarized documents. *Journal of the American society for information science and technology*, *54*(3), 203-215.

[19] Alzahrani, S., & Salim, N. (2010). Fuzzy semantic-based string similarity for extrinsic plagiarism detection. *Braschler and Harman*, *1176*, 1-8.

[20] Stamatatos, E. (2011). Plagiarism detection using stopword n-grams. *Journal of the American Society for Information Science and Technology*, *62*(12), 2512-2527.

[21] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130-137.

[22] Potthast, M., Stein, B., Barrón-Cedeño, A., & Rosso, P. (2010, August). An evaluation framework for plagiarism detection. In *Proceedings of the 23rd international conference on computational linguistics: Posters* (pp. 997-1005). Association for Computational Linguistics.

[23] Kent, C. K., & Salim, N. (2010, September). Web based cross language plagiarism detection. In *2010 Second International Conference on Computational Intelligence, Modelling and Simulation* (pp. 199-204). IEEE.

[24] Stamatatos, E. (2009). Intrinsic plagiarism detection using character n-gram profiles. *threshold*, *2*(1,500).

[25] Aiken, A. (1994). MOSS: A System for Detecting Software Similarity. Retrieved from http://theory.stanford.edu/~aiken/moss/ (Accessed June 11, 2019).

[26] Mason, T., Gavrilovska, A., & Joyner, D. A. (2019, February). Collaboration Versus Cheating: Reducing Code Plagiarism in an Online MS Computer Science Program. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1004-1010). ACM.

[27] Yan, L., McKeown, N., Sahami, M., & Piech, C. (2018, February). TMOSS: using intermediate assignment work to understand excessive collaboration in large classes. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 110-115). ACM.

[28] Mozgovoy, M., Fredriksson, K., White, D., Joy, M., & Sutinen, E. (2005, November). Fast plagiarism detection system. In *International Symposium on String Processing and Information Retrieval* (pp. 267-270). Springer, Berlin, Heidelberg.

[29] Bull, J., Colins, C., Coughlin, E., & Sharp, D. (2000). Technical review of plagiarism detection software report.

[30] Walker, J. (2010). Measuring plagiarism: Researching what students do, not what they say they do. *Studies in Higher Education*, *35*(1), 41-59.

[31] NLTK: Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.